ENT SENIOR DESIGN PROJECT REPORT

Automated and Monetized Ramen Noodle Vending Machine
Using Arduino Mega Microcontroller

Submitted to

Professor Goodman
Engineering Technology Department
by
Dan Ragozzino

November 7, 2018

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine

ABSTRACT

Contained in the following report is a complete and detailed document of the design and implementation for the development of an automated ramen noodle vending machine unit. Covered topics include justification and specification development, which considers target consumer audience as well as some considerations for safety ratings and features that are required for consumer appliances. Also covered is the development and implementation phase, entailing translating design into a physical and functional prototype that achieves specifications detailed in the design phase. There are three major components of the development phase: hardware, interface and software. Each component of development is covered in detail, including troubleshooting and on-the-fly changes made to design to accommodate issues that arise during the prototype development. The last section of the document includes some considerations for iterative design. Issues encountered during prototype development will also be documented. A fully detailed operations guide will also be included. In the appendices of the document are the full printout of the software, detailed spec sheets of individual hardware components, as well as documents from the design phase.

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine

TABLE OF CONTENTS

Contents

REVISION HISTORY

Version	Date	Revised by	Description
1.0	07 November 2018	Dan Ragozzino	Draft Version
1.5	14 November 2018	Dan Ragozzino	Refinement of all sections
1.6	26 November 2018	Dan Ragozzino	Refinement, and finalization

1. INTRODUCTION

The overarching purpose of this report is to fully document the design, development, production and testing of a prototype project for senior capstone project in the EET department at IUPUI for the Fall semester, 2018. The technical background for the ramen noodle capstone project lies within the development of wiring skills, programming knowledge, and electrical circuitry skills in both development and problem solving, all of which have been developed, grown, and scrutinized through an academic career at IUPUI. Within that scope, the ramen noodle vending is a marriage of hardware and software that allows for automation and control skills to be displayed. The focus on process automation is key for understanding the hardware design and development as well as the functional aspects of the software driving the hardware movement and operation. Throughout this document, each of the three major components of the project, hardware, software, and interface, will be explained from a design perspective, a development perspective, and from a testing and troubleshooting perspective. After the ramen noodle vending machine is fully explained, future considerations, troubleshooting guide, and recommendations will guide future development of the project.

1.1 Problem Statement & Justification

College students and in general people on the go are a challenging consumer audience to provide convenient and cost-effective food service to. To address this, particularly in the college campus setting, schools like IUPUI offer on the go food stations as well as established dining hall settings and off campus food offerings. However, these establishments suffer from some issues. One of the issues that is pertinent to college students is that these establishments aren't available all the time. When spending long hour on campus, food services may close at certain hours or not be open on weekends. Also, an issue to college students is that it might take time out of their focus and studies to go to the food services to get a meal, especially if the option is off campus. The solution to this that has been implemented on campuses is the availability of vending machines to provide round-the-clock food services to people looking for convenient offerings. The drawback to vending machines is often the types of products offered in their stock. Usually the limit is bags of chips or candy bars and sodas or bottled waters. There are some specialty machines that offer ice cream or fresh coffee, but they are specialty machines that have limited appeal.

To address the issues presented by the currently available options for food service, as well as satisfying the requirements for developing a prototype to fulfill the senior capstone project, proposed is a fully automated and self-contained vending machine that will make hot, ready to eat ramen noodle cups for customers. The vending machine form factor, like other vending machines, makes it possible for ramen noodles to be made anywhere, available any time for customers. A self-contained vending machine would be a good application of skills learned in ECET 30200, 37100, 30900, 30700 and IET 36400, as well as drawing on experiences in other 200 level electronics courses like ECET 20900, and 28400, as well as drawing on circuitry and simple programming skills, specifically state machine logic, learned in the basic courses like ECET 15700 and 15900. Vending machines are complex machines that operate on some of the fundamental principles that have been learned and experienced. Designing and developing a vending machine with the specialized function of delivering a readymade, hot cup of ramen would require programming, automation design, and user interface development, as well as basic electronic design and development.

1.2 Scope & Specifications

1.2.1 In Scope

Regarding scope of the project, the machine will provide a singular service to the end user. Within its operation, the machine will accept money and user selection as input, let the user know that their money and input are valid, and their selection is in stock, then proceed to pull the selection from the magazine, boil water, inject the water into the ramen cup, and then serve the cup, and a plastic fork, to the customer out the front of the machine. Within the scope is the time it takes from user input to delivery of the food, as well as the price that is set that the customer will need to get the product from the machine. Within scope is the temperature that the water is heated to and dispensed into the ramen cup. To ensure food safety, the water will need to be boiling or very near boiling when dispensed.

1.2.2 Out of Scope

Out of scope are the factors that are uncontrollable within the confines of the machine cabinet. Price of the product being stocked for the vendor, cost of electricity to run the machine, and cost of running water to the machine. Customer demand and usage of the machine as well as the real estate available to keep the machine are uncontrollable with respect to the operation of the machine. Also, out of scope is competition from other food vendors on or off campus that students and customers could go to alternatively.

1.3 System overview

The development of the vending machine is an involved process involving electrical, mechanical, digital, and analog elements, all culminating in a functional appliance that fulfills a

niche service to customers. The system will take user input and money, perform a task, and then produce a product/service for the user convenience per the specifications outlined. The project sponsor is Professor Elaine Cooney. Developer is Dan Ragozzino. Support agency is Indiana University – Purdue University of Indianapolis School of Engineering and Technology, EET Department. Users intended are IUPUI students and campus inhabitants. Due to non-compete contracts with the campus food vendors, machine is not to be set up for campus use, only private demonstration and testing.

2. REFERENCED DOCUMENTS

This section shall list the number, title, revision, and date of all documents referenced in this document. This section shall also identify the source for all documents not available through normal Government stocking activities.

Table 1: Reference Documents

No.	Reference Caption	Description	Appendix Item
1	16RlyLayout	Relay Module Board Layout / Schematic.	C.
2	PSU_Schematic	PCB layout schematic for built power supply	D.
3	BOM	Bill of Materials	E.
4	DC12_Motor	Spec sheet for DFRobot Fit 0492-A motor	F.
5	Linear_Actuator	Spec sheet for PA-07 micro linear Actuator	G.

3. SYSTEM-WIDE DESIGN DECISIONS

On a system wide basis, major components like motors and movement controls will interface using 12 volts DC power connections. Because of the power difference between Arduino I/O pins and the power requirements of the movement components, relays will be required to translate the outputs of the Arduino to workable signals to the motors. On a system wide basis, movements will be tracked with physical sensors. Some components, like proposed LCD and change acceptor will require serial communication with the Arduino board.

In conjunction with the scope of the project, power to the machine will be converted from 120 VAC using a modified ATX power supply. Customer payment options will be limited to only accepting quarters, dimes, and nickels. This is to streamline some of the internal operations, as accepting pennies and larger coins, like silver dollars, will complicate calculation and function of the machine.

3.1 Hardware Components

Hardware sketches for major component groups are included in Appendix item B.

Controller & Relays

The Arduino Mega 2560 is the main control board for the vending machine. The winning criteria for this board was the number of raw I/O available on the board without modification or additional hardware to expand, like some PLC modules. Arduino also adds an element of flexibility to the system, allowing for some things to be done that PLCs would have made complicated. For example, with the availability of 3.3 and 5 volts from the power supply, it would be possible to interface the Arduino Mega with internet of things components, expanding functionality of the machine to communicate with service technicians and to take credit cards or Jag Tags. This is in theory and not included in this version of the prototype. The use of the Arduino platform also taps into the massive library of open source materials, even to the extent of including examples for basic sketches in the IDE environment for Arduino. Libraries, examples, and assistance from the Arduino community are all available and easily found, another advantage over PLC development which would use proprietary software and be more difficult to find assistance for.

The Arduino is incapable of driving high voltage components, like many micro controllers. PLCs would have a distinct advantage of being able to drive 12VDC components natively, however it isn't a hopeless effort. Relay modules are developed specifically to work with Arduino platforms so that the low voltage and current control signals can run more powerful components like DC motors, which draw a lot of current and exceed the available voltage from the micro controller. The module chosen for this project is the SainSmart 12VDC, 16 module board (see document 16RlyLayout). The board has some built in features that allow it flexibility and capability for the vending machine. The board features optoisolators between the low voltage side and high voltage side, meaning that the Arduino is protected from possible damage from the high voltage components. The relay board requires its own 12VDC power to operate because of the number of relays on the module. The relay modules are active low devices, so they are initiated low in the Arduino code, which can be seen in the setup loop in Appendix item A.

Chassis

The chassis of the vending machine is an oversized erector set. Precut pieces of 12 AWG, angle steel with predrilled holes were used to construct the frame and support armatures of the cabinet. Corners are secured with 5/16x1/2-inch bolts and nuts. Overall dimensions of the cabinet are 6 feet tall, 2 feet wide, and 2 feet deep. There are four wooden shelves in the cabinet, each designated for a component group. The very bottom shelf is reserved for power and power distribution components. The power supplies and relay boards live on the very bottom. This was done as an effort to separate the power components from water handling components as much as possible. The next shelf up houses all the main movement components. Motors, feeding area, and delivery, along with all their tracking and positioning switches are all contained at this level, apart from the linear actuator. On the third shelf up is the bottom support of the feed tubes, the linear actuator, heating components, and water handling pumps and tubing. The top shelf serves mainly as the support for the top of the feed tubes, with the only electronic components up there being the stock input detection limit switches, mounted to the top opening of the feed tubes. Near the bottom of the feed tubes, about 6 inches up from the bottom, a servo motor is mounted to each tube. The

arm of the servo motor protrudes into the feed tube to hold the stack of cups, so the bottom cup can drop free to the feed area.

Power Supplies

There are two main power supply units in the cabinet. The first, and main power supply, is a modified EVGA 700-Watt ATX power supply that provides 12, 5, and 3.3 VDC outputs. For the prototype that this document covers, the 3.3-volt outputs are not used. The 700 W supply was chosen to handle all the motors and high current components being used for movement in the machine, as well as handle the current drawn by the relay module.

Second is a home-built power supply with three outputs. A 12VDC, 5VDC, and 120VAC line voltage. This power supply is built using linear regulators, a bridge rectifier, and step-down transformer to provide the dual dc outputs. See appendix item D for detailed schematic of power supply. The second function for this power supply was to provide an AC line voltage pass through with an inline fuse to protect from throwing a breaker, since the heater is a high power component.

Motors

Motors chosen for the prototype were chosen for their ease of interface, cost, and torque output. The DFRobot FIT 0492A (see appendix Item F). The low RPM made these motors ideal for the movement, allowing the mechanical components to move reasonable quick for their function. The torque output of these motors is also more than adequate for the prototype.

Linear Actuator

The Progressive Automation PA-07 Micro Linear Actuator (see appendix Item G) allows for simplified operation for the puncturing of the cup. Load capacity wasn't an issue for consideration since no other components would be hanging on the armature of the linear actuator.

Heater Core

The water heating element for the prototype was taken straight out of a Keurig coffee maker. It is a 120 VAC heating element. At 1460 Watts, the heater draws just over 12 Amps of current. The heater has included a mounted thermistor and a heat activated cutoff switch that will disable the heating element if the water gets above a designated temperature.

Switches & Controls

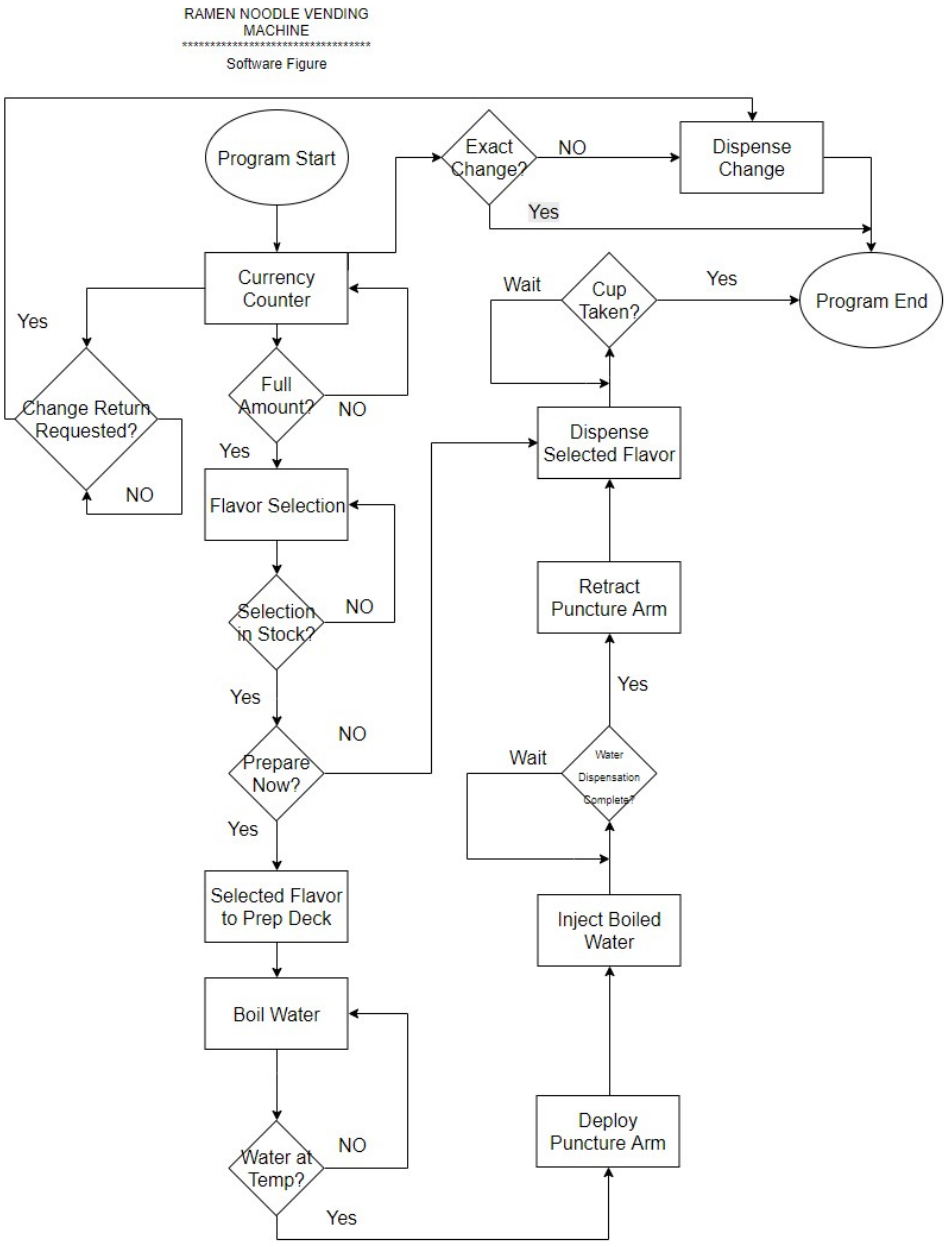
Movement and other operations, like restocking, are all tracked by micro limit switches. The limit switches interface with control surfaces on moving parts to allow the Arduino to track operations and movements.

Other Miscellaneous Components

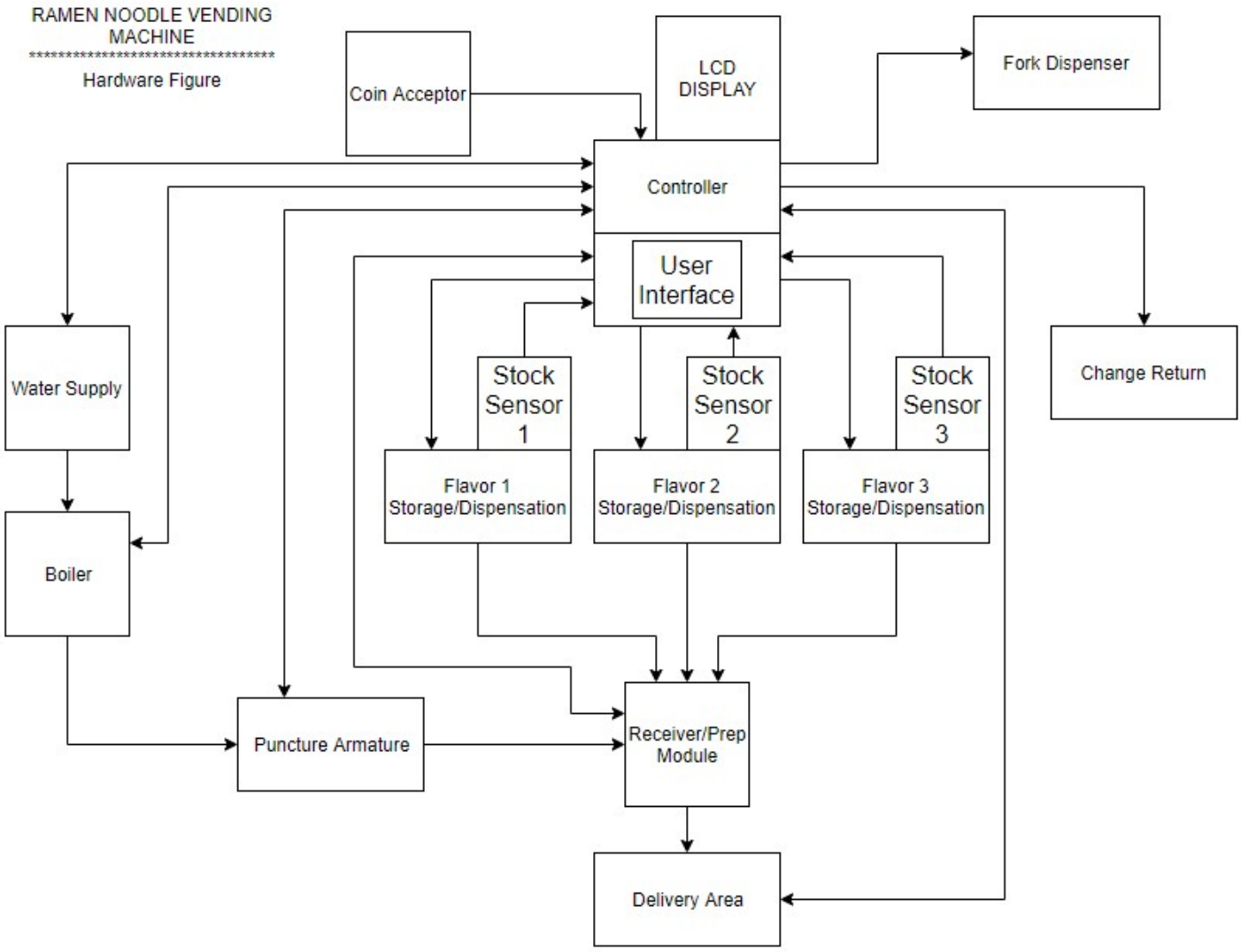
Hardware and miscellaneous connecting components were collected from a personal collection. Various thread pattern screws, washers, and connecting components were used to assemble components. See bill of materials in appendix item E for more information.

4. SYSTEM ARCHITECTURAL DESIGN

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine



Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine



4.1 System components

The most important component to this project is decidedly the controller that will manage and direct I/O for the process defined by the software flow chart and the hardware block diagram. Ultimately, the choice that was went with is the Arduino Mega 2560. The factors that contributed to the choice of this is the wide availability and compatibility of secondary market components like shields and adaptors for some interesting I/O devices, the overwhelming support from the opensource community, capability of native digital, analog, and serial communication without I/O expansions or adaptors, simplicity in communication setup and programming structure as well as flexibility when making changes to operations in software. While the Arduino is limited in some capacity with regards to graphical UI and some more sophisticated I/O devices that would have been adaptable with a device like a PLC, the cost savings afforded by the Arduino make mistakes more forgivable. Arduino also has the capability of expandability with serial communication, on top of the already impressive 54 I/O ports available onboard the Mega board.

The controller, being the brain of the machine, dictates some of the other components that are required for operation. Since the Arduino runs on 5VDC and many other components run on 12VDC, power availability is a concern that wouldn't be an issue with a PLC controller. Modifying a standard ATX power supply will provide enough amperage to power any of the components in the device and will allow the flexibility of the different voltage outputs being available, with multiple 12VDC channels, 5VDC channels, and 3.3VDC channels, as well as simply modified controls and safety features that will provide reliable and consistent power to the DC components. These factors make modifying a factory manufactured power supply ideal over developing a custom power supply unit and far superior to having separate power supply units for the different voltage requirements. The listed power supply, at 600W, has close to 50 Amps available on the +12 volt rail, allowing for plenty of power delivered to the DC motors, which max out at 2 Amps at locked rotor torque.

With the controller and power system decisions in mind, communicating the power and the controller is a process that will require some special hardware. The listed relay module has many relay channels that will be able to handle the number of I/O that use 12VDC. From the same manufacturer, there are other relay modules that are available in smaller sizes and with different features like serial communication.

For the movement systems in the machine, design choices are motivated by the principle of simplicity. The listed DC motor is capable enough to handle the different operations within the machine, with some hardware adaptation to translate the rotational movement into linear movement. Where rotational movements would be too bulky to translate, The linear actuator and solenoids listed will provide more precise movement capability to suit those operations, notably in the puncture arm extension and retraction and managing feeding noodle cups to the preparation area.

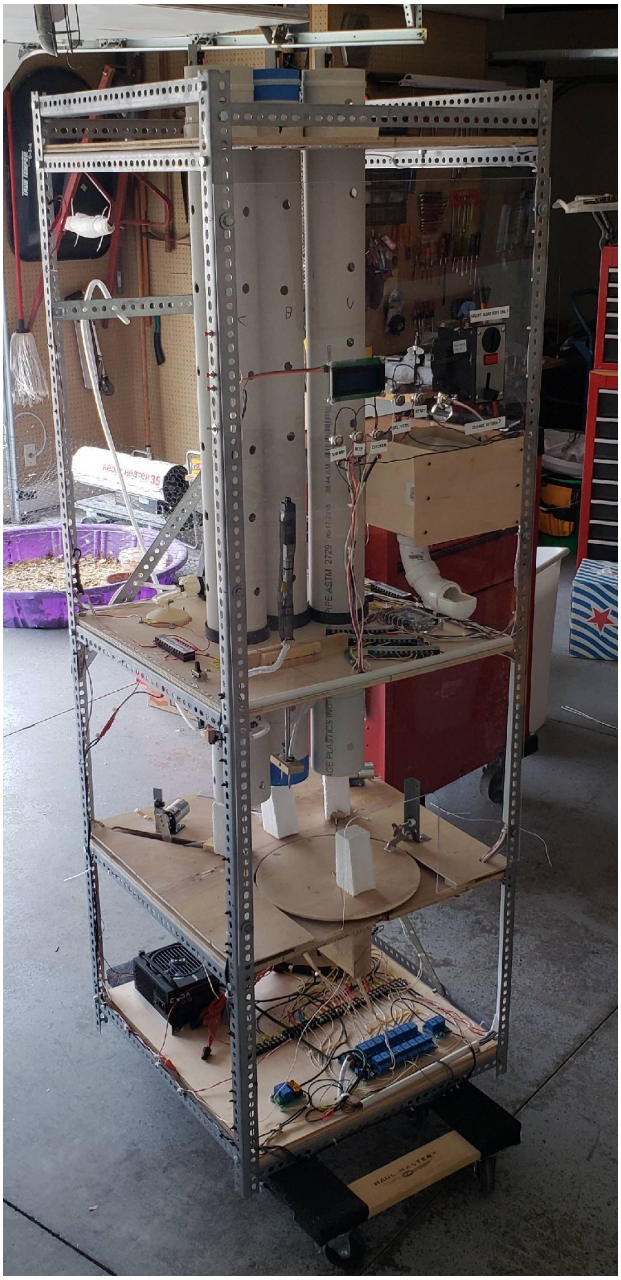
One of the main components for this machine regarding user interface is the ability of the machine to accept money to initiate the function of the machine. The chosen coin acceptor is capable of being programmed to detect, accept, and reject 6 different coin types. Some custom components will be required to develop a coin return mechanism that will provide the customer change for their purchase, however development of those components will involve simple parts like small DC motors like other movements in the machine.

Other components not listed in the parts list are some miscellaneous wires and connectors, the materials for the casing of the machine, the cups being used by the machine, and some custom components like the puncture needle and fabricated components to mount and adapt the motors and linear actuators to the movements in the machine.

Regarding some of the miscellaneous other components: push buttons, magnetic sensors, switches, and other UI components like the LCD and lighting, generic components will do well and will be able to be connected directly to the Arduino Mega board without the need for relay intermediaries. Magnetic sensors will be used to track rotational movement done by the motors and to keep track of position. Regarding the listed pressure transducer, this will be used to detect when a customer has taken their cup, signaling the end of the operational sequence in the program. It will also be used to detect if a cup has been placed on the machine correctly by the feeding components before the movement sequences to the puncture arm station. Some small components like resistors and fuses will be required to ensure safe wiring and reliable lighting of LEDs on the UI panel.

The last major functional component within the machine is the equipment designated to handle portioning, heating, and transferring water from a supply line, to a heating unit, then to the delivery needle. Rather than develop custom components to handle these operations, which lie at the heart of the operation of the machine, a small single serve coffee maker, like the listed Keurig, if found on the used market, will be an excellent source of some specialized components. The main component desired is the heating element and the level sensors that detect water portioning. The main motivation behind this choice is the safety ratings already attributed to the Keurig device, the time that the heating element takes to heat water to the defined temperature (192 degrees Fahrenheit, discovered through some research into the devices), and the simplicity in controlling the operation, which uses simple on/off controls, rather than proprietary serial controls.

4.2 Concept of execution



The software block diagram shows the lineage of decisions and how they relate to the hardware side of the machine. The software begins with an open-ended loop that waits for the user to input money. The machine will count coins until the required amount is met or exceeded. If the amount is exceeded, it will store that value for later. Note that at any point before the begin button is pressed, the user can cancel their order and have their change returned, however once the order is processed the user will not be able to get their money back. After money is counted, the machine waits for the user to indicate which flavor they want. The machine probes to see if that flavor is in stock and gives the user the option to choose another or cancel if they do not have their flavor in stock. The next decision is if the customer want their noodles made to order now or dispensed unfinished, for the user to prepare at their will later. if the user chooses to prepare later, the flavor will be dispensed unprepared and the order will be complete, and change will be returned after the cup is detected to be taken from the delivery area.

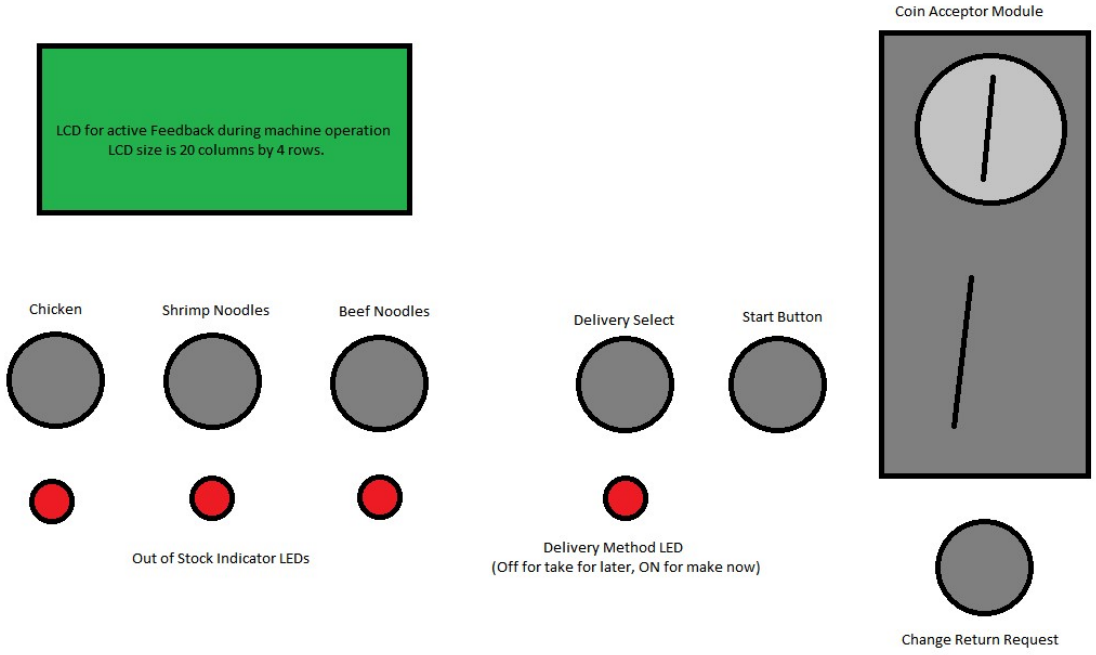
if the customer chooses to prepare now, the machine will branch to a series of functions to move the cup into position under a puncture the cup. When the cup is verified to be under position, the machine will dispense and heat a prescribed dosage of water. temperature sensors will monitor water temperature, and when it reaches near boiling (approx. 190 degrees Fahrenheit) the puncture arm will extend to puncture the top of the noodle cup. Once fully extended, water will be injected into the cup. Sensors will determine once all the water is dispensed and then retract the puncture tip. After retraction is complete, the cup will be moved to the delivery area. At this point the two decisions remerge into one line of function. The machine

will sense if the customer took the cup from the machine, dispense change and return to a waiting state until the next customer input is received.

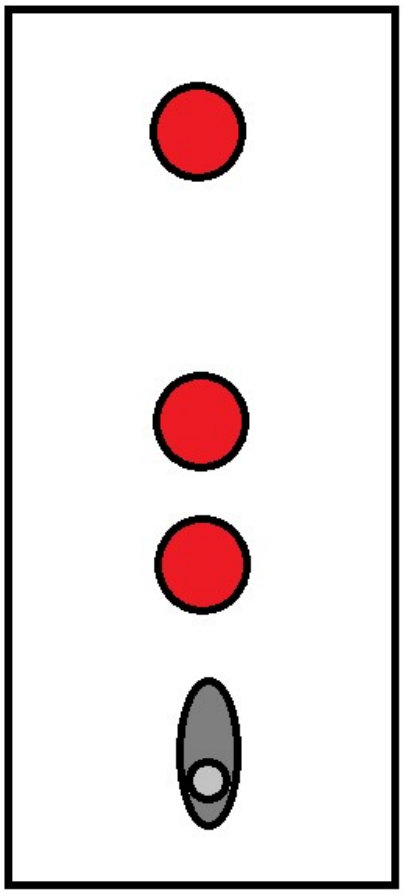
4.3 Interface design

In the below diagram in section 4.3.1 , from top left to bottom right, there will be an LCD display panel that will provide the customer with some feedback and instructions throughout the function of the machine. In its ready state there will be a welcome message that will prompt the user to insert coins to begin. To the right of the LCD panel are two push buttons that will have LED lighting to indicate which one will be active. Theses two buttons are mutually exclusive. They will indicate that the customer has chosen to make their noodles now or make their noodles later. To the right of those buttons, the rectangle represents the change intake unit. In the bottom left are the flavor selection buttons. The buttons themselves will have LED lighting to indicate selection. The small circle below will represent out of stock LED lights, that will indicate if that flavor is out of stock. Just right of the flavor selection buttons is the “begin” button. once proper change has been input, flavor selection made, and delivery method chosen, the user will press this button to tell the machine to make their noodles. To the right of that, below the change input, is a change return request button, which will have LED lighting to indicate when it will be available to be pressed.

4.3.1 Interface identification and diagrams



For Maintenance mode operation, a separate UI cluster exists on the side of the cabinet that will perform the cleaning operation, price adjustment, and operation mode selection.



4.3.2 User Setup and Operation

See Appendix Item H for complete operations manual.

Project Development Timeline

The original Gantt chart in appendix item J shows how project developed throughout the fall semester of 2018. From component purchasing and acquisition to troubleshooting and bug fixing, the original timeline mapped out how each phase of prototype development would go. However, during development, a lot of time consuming issues arose that delayed some elements in the Gantt chart. The result was delayed delivery and demonstration of the finished product by a full week beyond the original planned end date of the 17th of October. Prototype was demonstrated on the 24th of October.

5. CONCLUSIONS AND RECOMMENDATIONS

Prototype function after extensive testing and modifications to original designs and components was reasonably reliable in accordance with the design specification. Throughout testing, material choices became apparent problems regarding moving components. The shelving that movement components are attached to being made from thin plywood led to some slight warping that caused binding and movement issues. In future iterative designs alternative materials like metal or reinforced plexiglass would provide a more desirable structure that wouldn't be as susceptible. Another major design change through development was the inputs available for money. Original design called for the acceptance of multiple coins and the return of change if exact change was exceeded. Due to time constraints to meet deadlines for prototype delivery, and lack of experience dealing with Arduino programming and advanced functions, the choice to exclude the change return and limit change input to quarters only. For iterative changes advancing the prototype to accept a wider range of currency, like paper bills and credit cards, would be allowed by the flexibility of the Arduino and the built-in abilities of the ATX power supply. The availability of 3.3 volts and 5 volts allows for devices like Bluetooth modules and LTE connectivity to be added without adding complexity of the different voltage logic levels. Adding network connectivity could expand functionality to make the device integrate into the internet of things.

The last and most pressing issue presented by the prototype in its current state is the quality of the relay module and their interactions with the power supply. Unfortunately, some built in safety features of the power supply that protects the unit from voltage and current spikes, as well as sudden load changes, resulted in unintended supply shutdowns when relays were switched on. For iterative changes, a power supply without the protection features or relays with fly-back diodes across the coils to protect from sudden voltage or current spikes when the relay is closed. More refined movement elements and a redesigned stock management system to increase the stock capability of the machine would likely increase reliability and improve market viability by increasing the onboard stock, decreasing frequency of restocking.

NOTES

Arduino sketch loosely applies some principles and functions available in the example sketches in the Arduino IDE. Button debounce, input pullup, servo operation, and printing to the I2C LCD module are all principles learned and borrowing from the Arduino Examples, however, the deployment of the functions varies in the finished code.

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine

APPENDIXES

Item A . Full Arduino Code Printout. See comments for functional Notes. Comments denoted by // in front of comment string, or by a section comment with a /* at the beginning and */ at the end of the comment section.

```
/*
 * Author: Dan Ragozzino
 * Sketch Version: 7
 * Sketch Version Date: October 24, 2018
 *
 * Develpment Sponsors: Professor Elaine Cooney, ECET Dept, IUPUI
 *                      Professor Goodman, ECET Dept, IUPUI
 *
 * Development Purpose: Sketch is developed for Senior Capstone Project to drive
 *                      an automated ramen noodle vending machine. Developed for
 *                      ECET 49000 and ECET 49100.
 *
 * Sketch Summary:      Sketch will handle all inputs and outputs for function of
 *                      an automated ramen noodle vending machine. Inputs and Outputs
 *                      in hardware interface to allow for user input and automated
 *                      dispensing and preparation of ramen noodle cups.
 */
// Library inclusions, libraries are taken from open source
#include <LiquidCrystal_I2C.h> // required for communicating with the LCD display with the I2C backpack
#include <Wire.h> // required for some applications and communicating with serial components
#include <SoftwareSerial.h> // required for serial communication with change acceptor
#include <Servo.h> // required for driving servos

// delcares the dimensions of the lcd for later use
LiquidCrystal_I2C lcd(0x27, 20, 4);

// servo assignment, required to write servo positions to the servos later on
Servo stockSvo2;
Servo stockSvo1;
Servo stockSvo0;
Servo changeReturnSvo;

//variable declarations for user input buttons
byte flavorSelect0 = 19;
byte flavorSelect1 = 18;
byte flavorSelect2 = 17;
byte deliveryIndicator = 6;
byte deliverySelect = 2;
byte prepStart = 3;
byte mxMode = 16;
byte priceUp = 15;
byte priceDn = 14;
byte cleanStart = 4;
byte changeReturn = 5;
byte changeInhibit = 23;
byte changeInput = 0;

const int buzzer = 47;

// variable declarations for relay pins
const int inletWater = 24;
const int outletWater = 25;
const int feedMotor0 = 29;
const int feedMotor1 = 30;
const int feedMotor2 = 31;
const int mainMotorFwd = 26;
const int mainMotorRev = 27;
const int mainMotorGnd = 28;
const int linActDn = 32;
const int linActUp = 33;
const int linActGnd = 34;
const int waterHeater = 36;

// status used for delivery method toggling
int status = false;

// declarations for stock indicators
const int stockLight0 = 13;
const int stockLight1 = 12;
```

```
const int stockLight2 = 11;

const int stockIn0 = 9;
const int stockIn1 = 8;
const int stockIn2 = 7;

// position and tracking switch pin declarations
const int homePos = 35;
const int waterHandlePos = 37;
const int tubePos2 = 38;
const int tubePos1 = 39;
const int tubePos0 = 40;
const int motorSw2 = 41;
const int motorSw1 = 42;
const int motorSw0 = 43;


// switch case declarations, initilizes state cases to thier 0 state
byte mode = 0;
byte flavor = 0;
byte movement;
int delivery;
int changeCount = 0;

// some internal functional elements
int buttonState = 0;
int lastState = 0;
int count0 = 0;
int count1 = 0;
int count2 = 0;

int revCt0 = 1;
int revCt1 = 1;
int revCt2 = 1;

int stateUp = 0;
int stateDn = 0;
int lastStateUp = 0;
int lastStateDn = 0;
int coinRequired = 3;
volatile float pricePer = .75;
volatile float conversion;
////////////////////////////////////////////////////////////////////////////////////////
// END DECLRATION AND PRE CODE SETUP
////////////////////////////////////////////////////////////////////////////////////////
void setup() {
// serail initialzation and lcd initilization
Serial.begin(9600);
lcd.init();
lcd.backlight();
lcd.setCursor(2,1);
lcd.print("welcome");
delay(5000);
lcd.clear();
// PIN declarations and intitial state set for relays. (relays active LOW)
pinMode(inletWater, OUTPUT);
digitalWrite(inletWater,HIGH);
pinMode(outletWater, OUTPUT);
digitalWrite(outletWater,HIGH);
pinMode(feedMotor0, OUTPUT);
digitalWrite(feedMotor0,HIGH);
```

```
pinMode(feedMotor1, OUTPUT);
digitalWrite(feedMotor1,HIGH);
pinMode(feedMotor2, OUTPUT);
digitalWrite(feedMotor2,HIGH);
pinMode(mainMotorFwd, OUTPUT);
digitalWrite(mainMotorFwd,HIGH);
pinMode(mainMotorRev, OUTPUT);
digitalWrite(mainMotorRev,HIGH);
pinMode(mainMotorGnd, OUTPUT);
digitalWrite(mainMotorGnd,HIGH);
pinMode(linActDn, OUTPUT);
digitalWrite(linActDn,HIGH);
pinMode(linActUp, OUTPUT);
digitalWrite(linActUp,HIGH);
pinMode(linActGnd, OUTPUT);
digitalWrite(linActGnd,HIGH);
pinMode(waterHeater, OUTPUT);
digitalWrite(waterHeater, HIGH);
// PIN declarations for position and tracking switches
pinMode(homePos, INPUT);
digitalWrite(homePos, HIGH);
pinMode(waterHandlePos, INPUT);
digitalWrite(waterHandlePos, HIGH);
pinMode(tubePos2, INPUT);
digitalWrite(tubePos2, HIGH);
pinMode(tubePos1, INPUT);
digitalWrite(tubePos1, HIGH);
pinMode(tubePos0, INPUT);
digitalWrite(tubePos0 ,HIGH);
pinMode(motorSw2, INPUT);
digitalWrite(motorSw2, HIGH);
pinMode(motorSw1, INPUT);
digitalWrite(motorSw1, HIGH);
pinMode(motorSw0, INPUT);
digitalWrite(motorSw0, HIGH);
pinMode(stockIn0, INPUT);
digitalWrite(stockIn0, HIGH);
pinMode(stockIn1, INPUT);
digitalWrite(stockIn1, HIGH);
pinMode(stockIn2, INPUT);
digitalWrite(stockIn2, HIGH);
// PIN declarations and set to internal pullup for all UI buttons
pinMode(flavorSelect0, INPUT);
digitalWrite(flavorSelect0, HIGH);
pinMode(flavorSelect1, INPUT);
digitalWrite(flavorSelect1, HIGH);
pinMode(flavorSelect2, INPUT);
digitalWrite(flavorSelect2, HIGH);
pinMode(deliverySelect, INPUT);
digitalWrite(deliverySelect, HIGH);
pinMode(prepStart, INPUT);
digitalWrite(prepStart, HIGH);
pinMode(mxMode, INPUT);
digitalWrite(mxMode, HIGH);
pinMode(priceUp, INPUT);
digitalWrite(priceUp, HIGH);
pinMode(priceDn, INPUT);
digitalWrite(priceDn, HIGH);
pinMode(cleanStart, INPUT);
digitalWrite(cleanStart, HIGH);
pinMode(changeReturn, INPUT);
```

```
digitalWrite(changeReturn, HIGH);
// These two declarations are specific to the change inhibitor
pinMode(changeInhibit, OUTPUT);
digitalWrite(changeInhibit, LOW);
pinMode(changeInput, INPUT);
digitalWrite(changeInput, HIGH);
// stock indicator lights, declared outputs
pinMode(stockLight0, OUTPUT);
pinMode(stockLight1, OUTPUT);
pinMode(stockLight2, OUTPUT);
pinMode(deliveryIndicator, OUTPUT);
pinMode(buzzer, OUTPUT);
tone(buzzer, 1000);
delay(500);
noTone(buzzer);
// Servo attachments
stockSvo2.attach(44);\
stockSvo2.write(0);
stockSvo1.attach(45);
stockSvo1.write(50);
stockSvo0.attach(46);
stockSvo0.write(50);
changeReturnSvo.attach(10);
changeReturnSvo.write(0);

delay(500);
lcd.clear();
}
////////////////////////////////////////////////////////////////////////////////////////////
// END SETUP LOOP, ONLY RUNS ONCE ON RESET AND INITIAL POWER UP.
////////////////////////////////////////////////////////////////////////////////////////////
// Begin main program
void loop()
{
lcd.init();
Mx_Mode : // pointer to return to, referenced later on in code for use

// writes initial positions to servos.
stockSvo0.write(0);
stockSvo1.write(0);
stockSvo2.write(50);

// operations to be done while maintenance mode is switched on
while(digitalRead(mxMode) == LOW)
{
// from line 230 to 252 is for the price adjust up. each button press adds 25 cents to cost
stateUp = digitalRead(priceUp);

if(stateUp != lastStateUp)
{
  if(stateUp == LOW)
  {
    if(coinRequired < 10)
    {
      coinRequired++;
    }
    if(pricePer < 2.50)
    {
      pricePer += .25;
    }
  }
```

```
  else
  {
// do nothing
  }
  delay(20);
}
lastStateUp = stateUp;
////////////////////////////////////////////////////////
// from lines 253 to 275 is for the price adjust down, each button press decreases cost by 25 cents
stateDn = digitalRead(priceDn);

if(stateDn != lastStateDn)
{
  if(stateDn == LOW)
  {
    if(coinRequired > 2)
    {
      coinRequired--;
    }
    if(pricePer > .50)
    {
      pricePer -= .25;
    }
  }
  else
  {
// do nothing
  }
  delay(20);
}
lastStateDn = stateDn;

  // STOCK IS ONLY TO BE ADDED TO THE FEED TUBES IN MX MODE. WILL NOT ADJUST VALUE IF IN NORMAL OP.

  if(digitalRead(stockIn0) == LOW)
  {
    count0++;
    delay(250); //delay for debounce on switch
  }
  if(digitalRead(stockIn1) == LOW)
  {
    count1++;
    delay(250); //delay for debounce on switch
  }
  if(digitalRead(stockIn2) == LOW)
  {
    count2++;
    delay(250); //delay for debounce on switch
  }
  // prints the stock count for each of the three tubes
    lcd.setCursor(0,2);
    lcd.print(count2);
    lcd.setCursor(0,1);
    lcd.print(count1);
    lcd.setCursor(0,0);
    lcd.print(count0);

// BELOW is the code for running a cleaning operation.
// recommended cleaning regimen:
// clean every third or fourth restock.
// run two rounds of descaling solution (vinegar/water or designated descaling)
```

```
// !!!!! Make sure descaling fluid can be heated without issue.  !!!!!
// then run several rounds of regular water through the system to flush out cleaning solution.
// then return to normal op.
  if(digitalRead(cleanStart) == LOW)
  {
    lcd.setCursor(0,3);
    lcd.print("Clean Cycle Started");
    digitalWrite(inletWater, LOW);
    delay(10000);
    digitalWrite(waterHeater, LOW);
    delay(10000);
    digitalWrite(inletWater, HIGH);
    delay(10000);
    digitalWrite(waterHeater, HIGH);
    delay(20000);
    digitalWrite(outletWater, LOW);
    delay(30000);
    digitalWrite(outletWater, HIGH);
    lcd.setCursor(0, 3);
    lcd.print("Clean Cycle Complet");
  }
// Below is some indicators printed on the lcd showing inputted stock, and adjusted price per item.
  lcd.setCursor(11, 2);
  lcd.print(pricePer);
  changeCount = 0;
  flavor = 0;
  goto Mx_Mode; // refers to pointer at the top of the mx mode, loops back to mx mode beginning
}
stockSvo0.write(50);
stockSvo1.write(50);
stockSvo2.write(0);
goto Monetize; // goto statement to go to the change accepting state of the machine
Monetize : // pointer for going to change accpeting
lcd.clear();
lcd.setCursor(0,0);
lcd.print("INSERT QUARTERS ONLY");
lcd.setCursor(1,1);
lcd.print("Price:");
lcd.setCursor(1,2);
lcd.print(pricePer);
while(changeCount < coinRequired) // compares credit to required change
{
  lcd.setCursor(10,1);
  lcd.print("Credit:");
  digitalWrite(changeInhibit, HIGH);
  if(Serial.available() > 0)
  {
    if(Serial.peek() != -1)
    {
      changeCount += Serial.read();

      conversion = changeCount *.25;
      lcd.setCursor(10, 2);
      lcd.print(conversion);
      Serial.println(changeCount);
      Serial.println(" ");
    }
  }
}

prgmStart : // program pointer to return to the beginning of the code under certain conditions.
```

```
            // mainly if all stock is out or change is returned

////////////////////////////////////////////////////////////////////////////////////////
// SPECIAL NOTE: ALL SERIAL PRINT or PRINTLN OPERATIONS ARE FOR PROGRAMMING AND DEBUGGING ONLY, NOT ////
// USED IN NORMAL OPERATION, DISREGARD SERIAL OPERATIONS WHEN IN STANDARD OPERATION
lcd.clear();
lcd.setCursor(1,0);
lcd.print("select Flavor");
while(digitalRead(prepStart) == HIGH && changeCount >= coinRequired)
{
/*
 * switch case event used for user input, flavor selection, delivery method selection, and stock
 * indication.
 * in case 0, input pins are polled and mode is switched based on which input is high. Flavor is
 * switched to be used for later. If selected option is not in stock, the indicator light will
 * illuminate and a message will be displayed indicating selection is out of stock.
 *
 * After selection and delivery option is chosen, the start button exits the mode switch case
 * to go to the movement section of the code.
 */
  switch(mode)
  {
    case 0:
    digitalWrite(changeInhibit, HIGH);
        if(mode == 0 && digitalRead(mxMode) == LOW)
    {
      lcd.init();
      lcd.clear();
      lcd.setCursor(3, 0);
      lcd.print(": stk 0");
      lcd.setCursor(3,1);
      lcd.print(": stk 1");
      lcd.setCursor(3,2);
      lcd.print(": stk 2");
      lcd.setCursor( 10, 1);
      lcd.print( " Prc Per:");
      goto Mx_Mode;
    }
    if(count0 != 0){digitalWrite(stockLight0, LOW);}
    if(count1 != 0){digitalWrite(stockLight1, LOW);}
    if(count2 != 0){digitalWrite(stockLight2, LOW);}
    if(mode == 0 && digitalRead(flavorSelect0) == LOW && count0 != 0){mode = 1; flavor = 1;}
    if(mode == 0 && digitalRead(flavorSelect0) == LOW && count0 == 0){mode = 2; flavor = 0;}
    if(mode == 0 && digitalRead(flavorSelect1) == LOW && count1 != 0){mode = 3; flavor = 2;}
    if(mode == 0 && digitalRead(flavorSelect1) == LOW && count1 == 0){mode = 4; flavor = 0;}
    if(mode == 0 && digitalRead(flavorSelect2) == LOW && count2 != 0){mode = 5; flavor = 3;}
    if(mode == 0 && digitalRead(flavorSelect2) == LOW && count2 == 0){mode = 6; flavor = 0;}
    if(mode == 0 && digitalRead(changeReturn) == LOW){mode = 7; flavor = 0;}
//*****************************************************************************************// this
section of code toggles the delivery options, by default delivery is zero(0), the state
// for making the noodles now. Pressing the button will extinguish the delivery button, meaning
// the user has selected to take the noodles dry for later.
  if(digitalRead(deliverySelect) == LOW)
  {
    status = !status;
    delivery = status;
    Serial.println(status);
    digitalWrite(deliveryIndicator, status);
    tone(buzzer, 2000);
    delay(500);
    noTone(buzzer);
```

```
  }
  while(digitalRead(deliverySelect) == LOW)
  {
    delay(50);
  }
    break;
//*********************************************************************************************
    case 1: // selection state if chicken is selected and is in stock
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Chicken Noodles    ");
    Serial.println(flavor);
    delay(100);
    mode = 0;
    tone(buzzer, 1200);
    delay(250);
    noTone(buzzer);
    break;
//////////////////////////////////////////////////////////
    case 2: // selection state if chicken is selected but is out of stock
    digitalWrite(stockLight0, HIGH);
    lcd.setCursor(0,0);
    lcd.print("chicken out of stock");
    delay(100);
    tone(buzzer, 1000);
    delay(100);
    noTone(buzzer);
    delay(100);
    tone(buzzer, 1000);
    delay(100);
    noTone(buzzer);
    mode = 0;
    break;
//////////////////////////////////////////////////////////
    case 3: // selection state if beef is selected and is in stock
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Beef Noodles        ");
    Serial.println(flavor);
    tone(buzzer, 1800);
    delay(250);
    noTone(buzzer);
    delay(100);
    mode = 0;
    break;
//////////////////////////////////////////////////////////
    case 4: // selection state if beef is selected but is out of stock
    digitalWrite(stockLight1, HIGH);
    lcd.setCursor(0,0);
    lcd.print("beef out of stock   ");
    delay(100);
    tone(buzzer, 1000);
    delay(100);
    noTone(buzzer);
    delay(100);
    tone(buzzer, 1000);
    delay(100);
    noTone(buzzer);
    mode = 0;
    break;
//////////////////////////////////////////////////////////
```

```
    case 5: // selection state if shirmp is selected and is in stock
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Veggie Noodles      ");
    Serial.println(flavor);
    delay(100);
    tone(buzzer, 1500);
    delay(250);
    noTone(buzzer);
    mode = 0;
    break;
////////////////////////////////////////////////////////
    case 6: // selection state if shrimp is selected but is out of stock
    digitalWrite(stockLight2, HIGH);
    lcd.setCursor(0,0);
    lcd.print("veggie out of stock ");
    delay(100);
    tone(buzzer, 1000);
    delay(100);
    noTone(buzzer);
    delay(100);
    tone(buzzer, 1000);
    delay(100);
    noTone(buzzer);
    mode = 0;
    break;
////////////////////////////////////////////////////////
    case 7: // case to return to change acceptance
    mode = 0;
    changeCount = 0;
    movement = 0;
    goto Monetize;
    break;
////////////////////////////////////////////////////////
  }
//////////////////////////////////////////////////////////////////////////////////////
// BELOW THIS ARE INSTRUCTIONS FOR HARDWARE MOVEMENT AND TRACKING. DO NOT EDIT, DO NOT ALTER, DO NOT
ADJUST TIMING ////////////////////////////////////////////////////////////////////////
/*
 * If special Flavors are stocked in machine, request the tags and displays be altered from the factory
 * DO NOT edit lcd display strings, serial outputs, or tags for operation.
 */
}

// This section of the code exits the switch case waiting for user input.
if(digitalRead(prepStart) == LOW && flavor != 0)
{
  Serial.println("Start initiated");
  if(status == true){lcd.setCursor(2,1); lcd.print("make now      ");}
  if(status == false){lcd.setCursor(1,1); lcd.print("take for later");}
  tone(buzzer, 1200);
  delay(250);
  noTone(buzzer);
  //Serial.println(flavor);
  //Serial.println(status);
  //Serial.println(delivery);
  delay(250);
// decision for movement based on conditions for flavor and delivery selections
  if(flavor == 1 && delivery == 1){movement = 1;}
  if(flavor == 1 && delivery == 0){movement = 2;}
  if(flavor == 2 && delivery == 1){movement = 3;}
```

```
  if(flavor == 2 && delivery == 0){movement = 4;}
  if(flavor == 3 && delivery == 1){movement = 5;}
  if(flavor == 3 && delivery == 0){movement = 6;}
  Serial.println(movement);
  switch(movement)
  {
    case 0: // initial movement state, not used in normal operation
// do nothing
    break;
//*********************************************************************************************
    case 1: // movement case for dispensing hot chicken noodles
    lcd.setCursor(0,1);
    lcd.print("making Noodles");
    do
    {
      digitalWrite(mainMotorRev, LOW);
    }while(digitalRead(tubePos0) == HIGH);

    digitalWrite(mainMotorRev, HIGH);
    delay(10);
    stockSvo0.write(0);
    delay(1000);
    stockSvo0.write(50);
    delay(1500);
    while(revCt0 > 0)
    {
      if(digitalRead(tubePos0) == LOW)
      {
        digitalWrite(feedMotor0, LOW);
      }
      if(digitalRead(motorSw0) == LOW)
      {
        digitalWrite(feedMotor0, HIGH);
        revCt0 -= 1;
      }
    }
    delay(20);
    do
    {
      digitalWrite(mainMotorFwd, LOW);
      digitalWrite(mainMotorGnd, LOW);
    }while(digitalRead(waterHandlePos) == HIGH);

    if(digitalRead(waterHandlePos) == LOW);
    {
      digitalWrite(mainMotorFwd, HIGH);
      digitalWrite(mainMotorGnd, HIGH);
      delay(25);
      digitalWrite(linActGnd, LOW);
      digitalWrite(linActDn, LOW);
      delay(10000);
      digitalWrite(linActGnd, HIGH);
      digitalWrite(linActDn, HIGH);
      delay(25);
      digitalWrite(inletWater, LOW);
      delay(12000);
      digitalWrite(inletWater, HIGH);
digitalWrite(linActGnd,HIGH);
digitalWrite(linActUp,HIGH);
digitalWrite(linActDn,HIGH);
digitalWrite(mainMotorGnd,HIGH);
```

```
digitalWrite(mainMotorRev,HIGH);
digitalWrite(mainMotorFwd,HIGH);
digitalWrite(feedMotor2,HIGH);
digitalWrite(feedMotor1,HIGH);
digitalWrite(feedMotor0,HIGH);
digitalWrite(outletWater,HIGH);
digitalWrite(inletWater,HIGH);
digitalWrite(waterHeater, HIGH);
      delay(60000);
      digitalWrite(outletWater, LOW);
      delay(25000);
      digitalWrite(outletWater, HIGH);
      delay(2000);
      digitalWrite(linActUp, LOW);
      delay(9000);
      digitalWrite(linActUp, HIGH);
    }
    do
    {
      digitalWrite(mainMotorGnd, LOW);
      digitalWrite(mainMotorFwd, LOW);
    }while(digitalRead(homePos)== HIGH);
    digitalWrite(mainMotorFwd, HIGH);
    digitalWrite(mainMotorGnd, HIGH);
    revCt0 = 1;
    lcd.setCursor(0, 1);
    lcd.print("Take Noodles Below");
    delay(2000);
    lcd.clear();
    //Serial.println("dispensed HOT chicken");
    if(count0 > 0)
    {
      count0 -= 1;
    }
    tone(buzzer, 2000);
    delay(100);
    noTone(buzzer);
    delay(100);
    tone(buzzer, 2000);
    delay(100);
    noTone(buzzer);
    changeCount = 0;
    digitalWrite(changeInhibit, LOW);
    delay(500);
    lcd.clear();
    break;
//*********************************************************************************************
    case 2: // movement case for dispensing cold chicken noodles
    lcd.setCursor(0,1);
    lcd.print("getting Noodles");
    do
    {
      digitalWrite(mainMotorRev, LOW);
    }while(digitalRead(tubePos0) == HIGH);
    digitalWrite(mainMotorRev, HIGH);
    delay(10);
    stockSvo0.write(0);
    delay(1000);
    stockSvo0.write(50);
    delay(1500);
    while(revCt0 > 0)
```

```
    {
      if(digitalRead(tubePos0) == LOW)
      {
        digitalWrite(feedMotor0, LOW);
      }
      if(digitalRead(motorSw0) == LOW)
      {
        digitalWrite(feedMotor0, HIGH);
        revCt0 -= 1;
      }
    }
    delay(20);
        do
    {
      digitalWrite(mainMotorGnd, LOW);
      digitalWrite(mainMotorFwd, LOW);
    }while(digitalRead(homePos)== HIGH);
    digitalWrite(mainMotorFwd, HIGH);
    digitalWrite(mainMotorGnd, HIGH);

    revCt0 = 1;
    lcd.setCursor(0, 1);
    lcd.print("Take Noodles Below");
    delay(2000);
    lcd.clear();
    //Serial.println("dispensed COLD chicken");
    if(count0 > 0)
    {
      count0 -= 1;
    }
    tone(buzzer, 2000);
    delay(100);
    noTone(buzzer);
    delay(100);
    tone(buzzer, 2000);
    delay(100);
    noTone(buzzer);
    changeCount = 0;
    digitalWrite(changeInhibit, LOW);
    delay(500);
    lcd.clear();
    break;
//*********************************************************************************************
    case 3: // movement case for dispensing hot beef noodles
    lcd.setCursor(0,1);
    lcd.print("making Noodles");
    do
    {
      digitalWrite(mainMotorRev, LOW);
    }while(digitalRead(tubePos1) == HIGH);
    digitalWrite(mainMotorRev, HIGH);
    delay(10);
    stockSvo1.write(50);
    delay(1000);
    stockSvo1.write(0);
    delay(1500);
    while(revCt1 > 0)
    {
      if(digitalRead(tubePos1) == LOW)
      {
        digitalWrite(feedMotor1, LOW);
```

```
      }
      if(digitalRead(motorSw1) == LOW)
      {
        digitalWrite(feedMotor1, HIGH);
        revCt1 -= 1;
      }
    }
    delay(20);
    do
    {
      digitalWrite(mainMotorFwd, LOW);
      digitalWrite(mainMotorGnd, LOW);
    }while(digitalRead(waterHandlePos) == HIGH);
    if(digitalRead(waterHandlePos) == LOW);
    {
      digitalWrite(mainMotorFwd, HIGH);
      digitalWrite(mainMotorGnd, HIGH);
      delay(25);
      digitalWrite(linActGnd, LOW);
      digitalWrite(linActDn, LOW);
      delay(10000);
      digitalWrite(linActGnd, HIGH);
      digitalWrite(linActDn, HIGH);
      delay(25);
      digitalWrite(inletWater, LOW);
      delay(12000);
      digitalWrite(inletWater, HIGH);
digitalWrite(linActGnd,HIGH);
digitalWrite(linActUp,HIGH);
digitalWrite(linActDn,HIGH);
digitalWrite(mainMotorGnd,HIGH);
digitalWrite(mainMotorRev,HIGH);
digitalWrite(mainMotorFwd,HIGH);
digitalWrite(feedMotor2,HIGH);
digitalWrite(feedMotor1,HIGH);
digitalWrite(feedMotor0,HIGH);
digitalWrite(outletWater,HIGH);
digitalWrite(inletWater,HIGH);
digitalWrite(waterHeater, HIGH);
      delay(60000);
      digitalWrite(outletWater, LOW);
      delay(25000);
      digitalWrite(outletWater, HIGH);
      delay(2000);
      digitalWrite(linActUp, LOW);
      delay(9000);
      digitalWrite(linActUp, HIGH);
    }
    do
    {
      digitalWrite(mainMotorGnd, LOW);
      digitalWrite(mainMotorFwd, LOW);
    }while(digitalRead(homePos)== HIGH);
    digitalWrite(mainMotorFwd, HIGH);
    digitalWrite(mainMotorGnd, HIGH);
    revCt1 = 1;
    lcd.setCursor(0, 1);
    lcd.print("Take Noodles Below");
    delay(2000);
    lcd.clear();
    //Serial.println("dispensed HOT beef");
```

```
    if(count1 > 0)
    {
      count1 -= 1;
    }
    tone(buzzer, 2000);
    delay(100);
    noTone(buzzer);
    delay(100);
    tone(buzzer, 2000);
    delay(100);
    noTone(buzzer);
    changeCount = 0;
    digitalWrite(changeInhibit, LOW);
    delay(500);
    lcd.clear();
    break;
//*********************************************************************************************
    case 4: // movement case for dispensing cold beef noodles
    lcd.setCursor(0,1);
    lcd.print("getting Noodles");
    do
    {
      digitalWrite(mainMotorRev, LOW);
    }while(digitalRead(tubePos1) == HIGH);

    digitalWrite(mainMotorRev, HIGH);
    delay(10);
    stockSvo1.write(0);
    delay(1000);
    stockSvo1.write(50);
    delay(1500);
    while(revCt1 > 0)
    {
      if(digitalRead(tubePos1) == LOW)
      {
        digitalWrite(feedMotor1, LOW);
      }
      if(digitalRead(motorSw1) == LOW)
      {
        digitalWrite(feedMotor1, HIGH);
        revCt1 -= 1;
      }
    }
    delay(20);
        do
    {
      digitalWrite(mainMotorGnd, LOW);
      digitalWrite(mainMotorFwd, LOW);
    }while(digitalRead(homePos)== HIGH);

    digitalWrite(mainMotorFwd, HIGH);
    digitalWrite(mainMotorGnd, HIGH);
    revCt1 = 1;
    lcd.setCursor(0, 1);
    lcd.print("Take Noodles Below");
    delay(2000);
    lcd.clear();
    //Serial.println("dispensed COLD beef");
    if(count1 > 0)
    {
      count1 -= 1;
```

```
    }
    tone(buzzer, 2000);
    delay(100);
    noTone(buzzer);
    delay(100);
    tone(buzzer, 2000);
    delay(100);
    noTone(buzzer);
    changeCount = 0;
    digitalWrite(changeInhibit, LOW);
    delay(500);
    lcd.clear();
    break;
//*********************************************************************************************
    case 5: // movement case for dispensing hot shrimp noodles
    lcd.setCursor(0,1);
    lcd.print("making Noodles");
    do
    {
      digitalWrite(mainMotorRev, LOW);
    }while(digitalRead(tubePos2) == HIGH);
    digitalWrite(mainMotorRev, HIGH);
    delay(10);
    stockSvo2.write(50);
    delay(1000);
    stockSvo2.write(0);
    delay(1500);
    while(revCt2 > 0)
    {
      if(digitalRead(tubePos2) == LOW)
      {
        digitalWrite(feedMotor2, LOW);
      }
      if(digitalRead(motorSw2) == LOW)
      {
        digitalWrite(feedMotor2, HIGH);
        revCt2 -= 1;
      }
    }
    delay(20);
    do
    {
      digitalWrite(mainMotorFwd, LOW);
      digitalWrite(mainMotorGnd, LOW);
    }while(digitalRead(waterHandlePos) == HIGH);
    if(digitalRead(waterHandlePos) == LOW);
    {
      digitalWrite(mainMotorFwd, HIGH);
      digitalWrite(mainMotorGnd, HIGH);
      delay(25);
      digitalWrite(linActGnd, LOW);
      digitalWrite(linActDn, LOW);
      delay(10000);
      digitalWrite(linActGnd, HIGH);
      digitalWrite(linActDn, HIGH);
      delay(25);
      digitalWrite(inletWater, LOW);
      delay(12000);
      digitalWrite(inletWater, HIGH);
digitalWrite(linActGnd,HIGH);
digitalWrite(linActUp,HIGH);
```

```
digitalWrite(linActDn,HIGH);
digitalWrite(mainMotorGnd,HIGH);
digitalWrite(mainMotorRev,HIGH);
digitalWrite(mainMotorFwd,HIGH);
digitalWrite(feedMotor2,HIGH);
digitalWrite(feedMotor1,HIGH);
digitalWrite(feedMotor0,HIGH);
digitalWrite(outletWater,HIGH);
digitalWrite(inletWater,HIGH);
digitalWrite(waterHeater, HIGH);

      delay(60000);


      digitalWrite(outletWater, LOW);
      delay(25000);
      digitalWrite(outletWater, HIGH);

      delay(2000);

      digitalWrite(linActUp, LOW);
      delay(9000);
      digitalWrite(linActUp, HIGH);
    }

    do
    {
      digitalWrite(mainMotorGnd, LOW);
      digitalWrite(mainMotorFwd, LOW);
    }while(digitalRead(homePos)== HIGH);

    digitalWrite(mainMotorFwd, HIGH);
    digitalWrite(mainMotorGnd, HIGH);

    revCt1 = 1;
    lcd.setCursor(0, 1);
    lcd.print("Take Noodles Below");
    delay(2000);
    lcd.clear();
    Serial.println("dispensed COLD Veggie");
    if(count2 > 0)
    {
      count2 -= 1;
    }
    tone(buzzer, 2000);
    delay(100);
    noTone(buzzer);
    delay(100);
    tone(buzzer, 2000);
    delay(100);
    noTone(buzzer);
    changeCount = 0;
    digitalWrite(changeInhibit, LOW);
    delay(500);
    lcd.clear();
    break;
//*************************************************************************************************
    case 6: // movement case for dispensing cold shrimp noodles
    lcd.setCursor(0,1);
    lcd.print("getting Noodles");
    do
```

```
    {
      digitalWrite(mainMotorRev, LOW);
    }while(digitalRead(tubePos2) == HIGH);
    digitalWrite(mainMotorRev, HIGH);
    delay(10);
    stockSvo2.write(50);
    delay(1000);
    stockSvo2.write(0);
    delay(1500);
    while(revCt2 > 0)
    {
      if(digitalRead(tubePos2) == LOW)
      {
        digitalWrite(feedMotor2, LOW);
      }
      if(digitalRead(motorSw2) == LOW)
      {
        digitalWrite(feedMotor2, HIGH);
        revCt2 -= 1;
      }
    }
    delay(20);
        do
    {
      digitalWrite(mainMotorGnd, LOW);
      digitalWrite(mainMotorFwd, LOW);
    }while(digitalRead(homePos)== HIGH);

    digitalWrite(mainMotorFwd, HIGH);
    digitalWrite(mainMotorGnd, HIGH);
    revCt2 = 1;
    lcd.setCursor(0, 1);
    lcd.print("Take Noodles Below");
    delay(2000);
    lcd.clear();
    Serial.println("dispensed COLD Veggie");
    if(count2 > 0)
    {
      count2 -= 1;
    }
    tone(buzzer, 2000);
    delay(100);
    noTone(buzzer);
    delay(100);
    tone(buzzer, 2000);
    delay(100);
    noTone(buzzer);
    changeCount = 0;
    digitalWrite(changeInhibit, LOW);
    delay(500);
    lcd.clear();
    break;
   }
  goto Monetize;
}
}
    END  OF  MAIN  PROGRAM
```

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine

Item B. Initial Sketches of Hardware Modules.

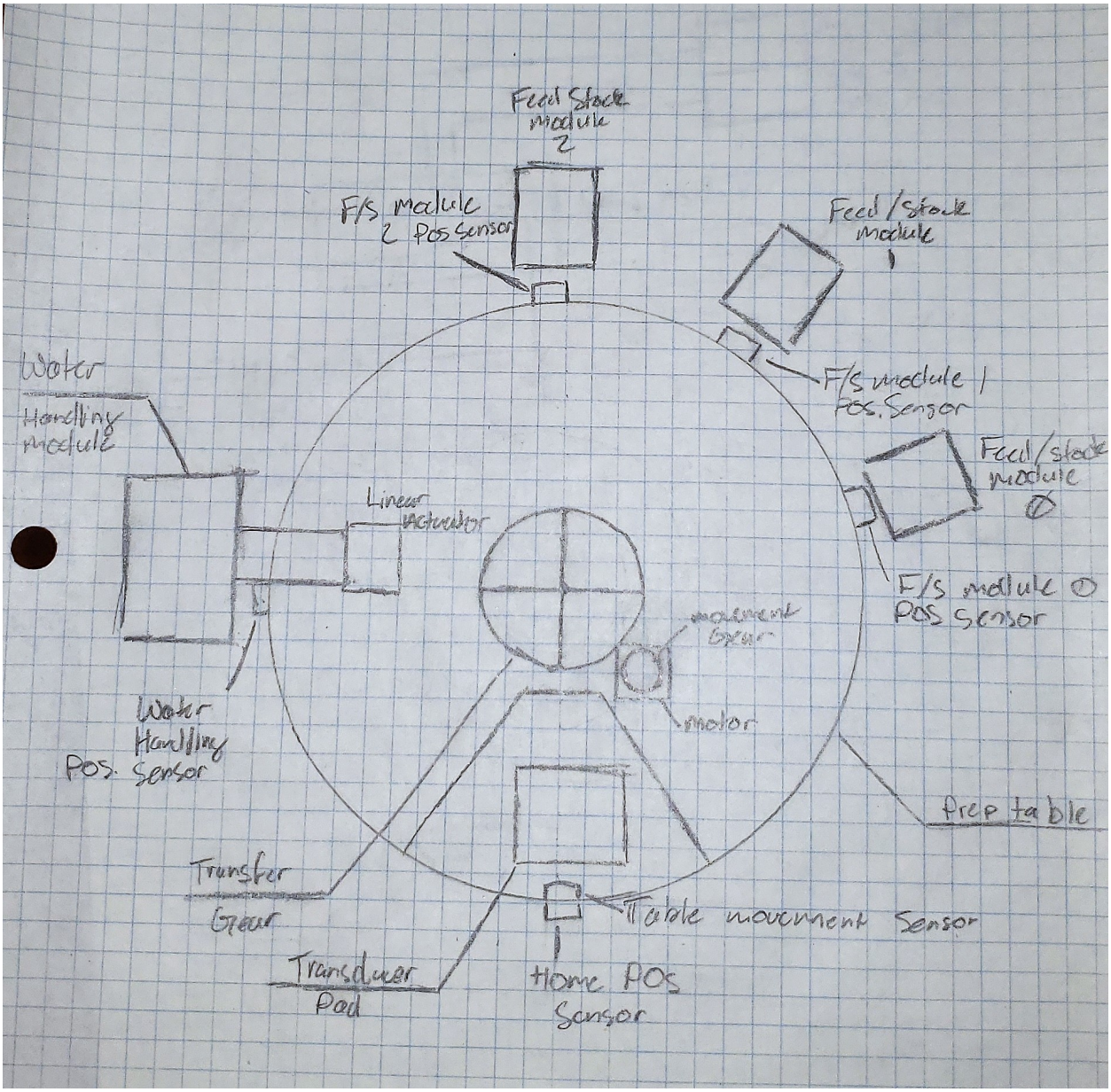


Figure 1, Rough Sketch for Preparation Tray in Cabinet

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine

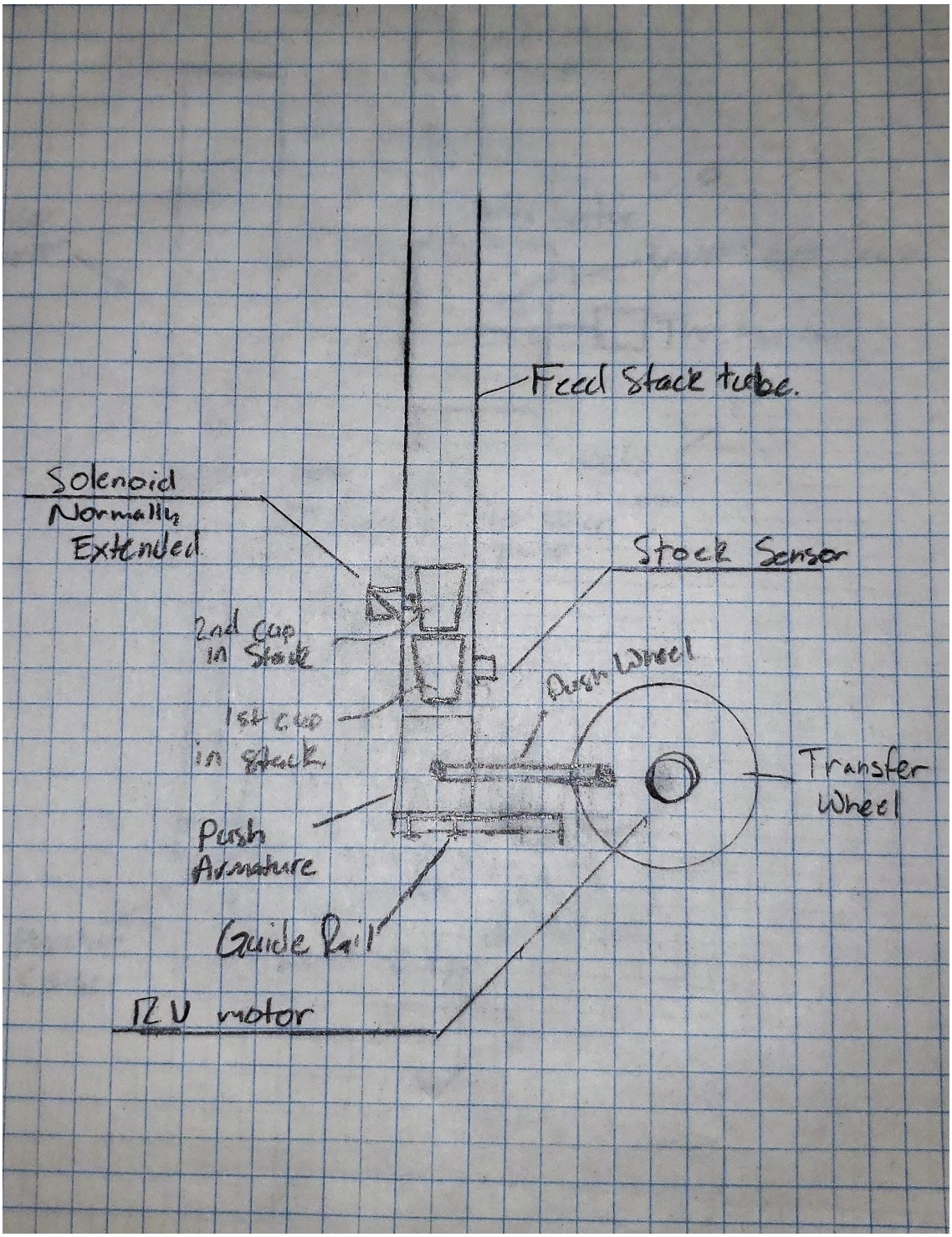


Figure 2, Rough Sketch for Feed Tubes

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine

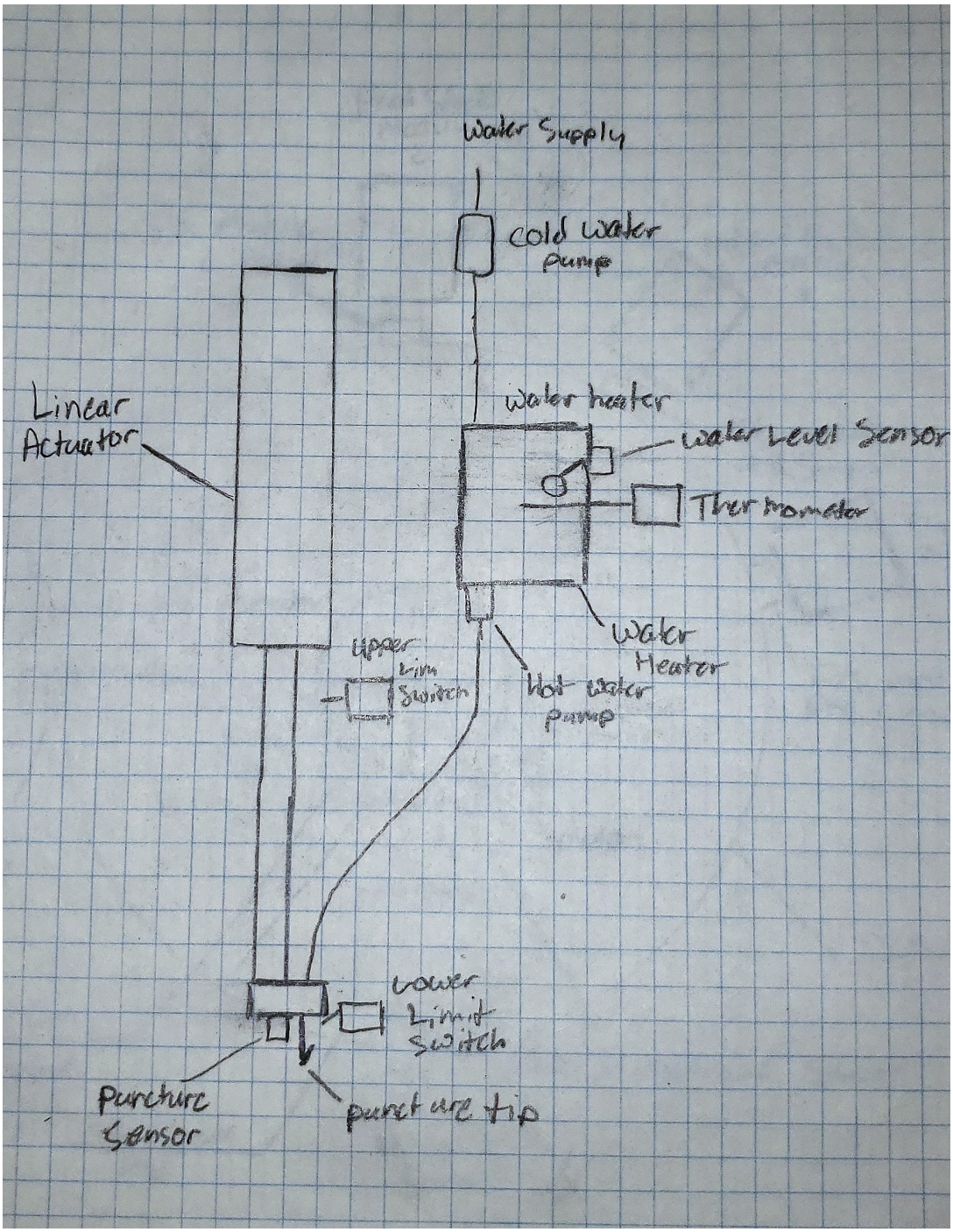
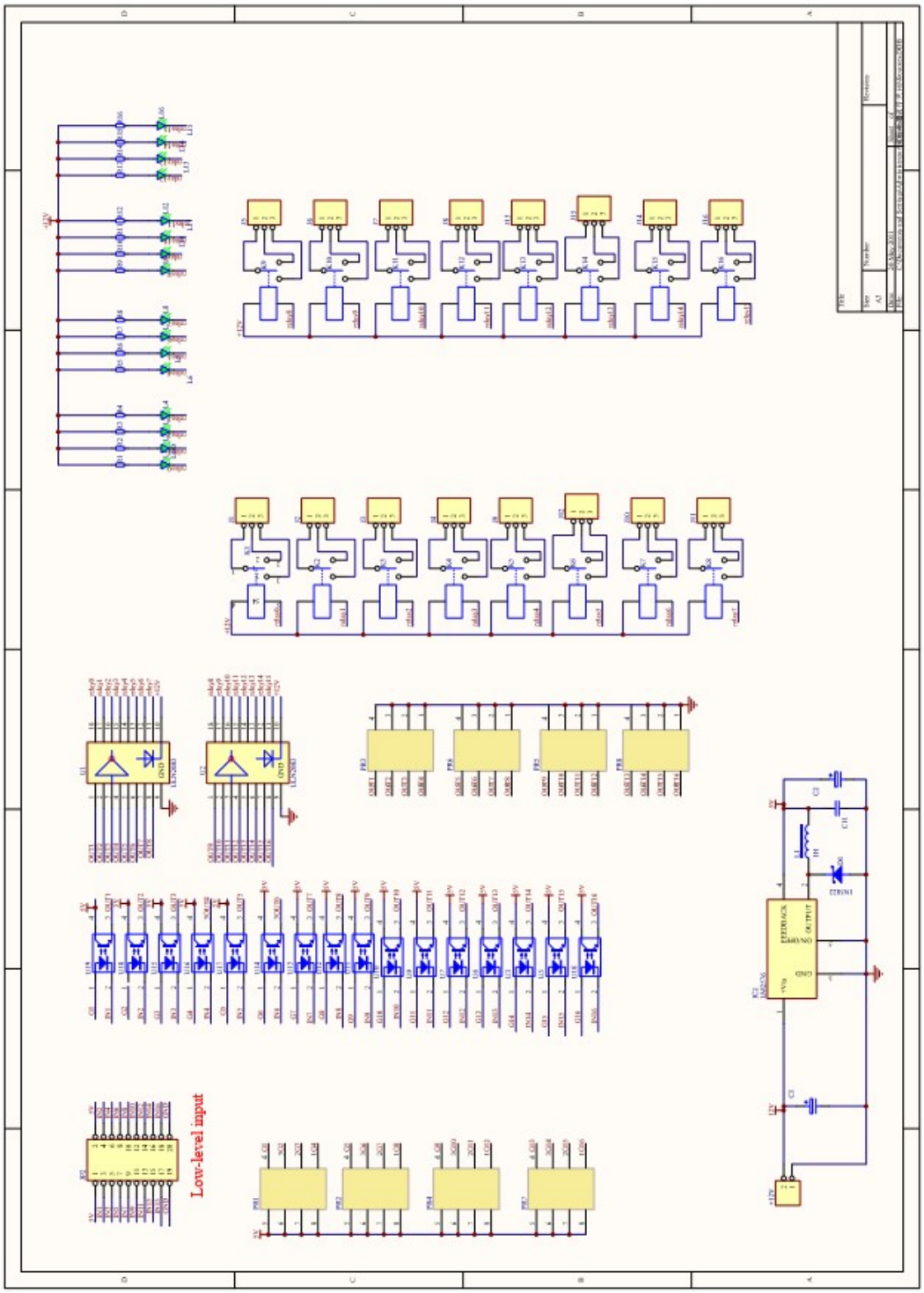


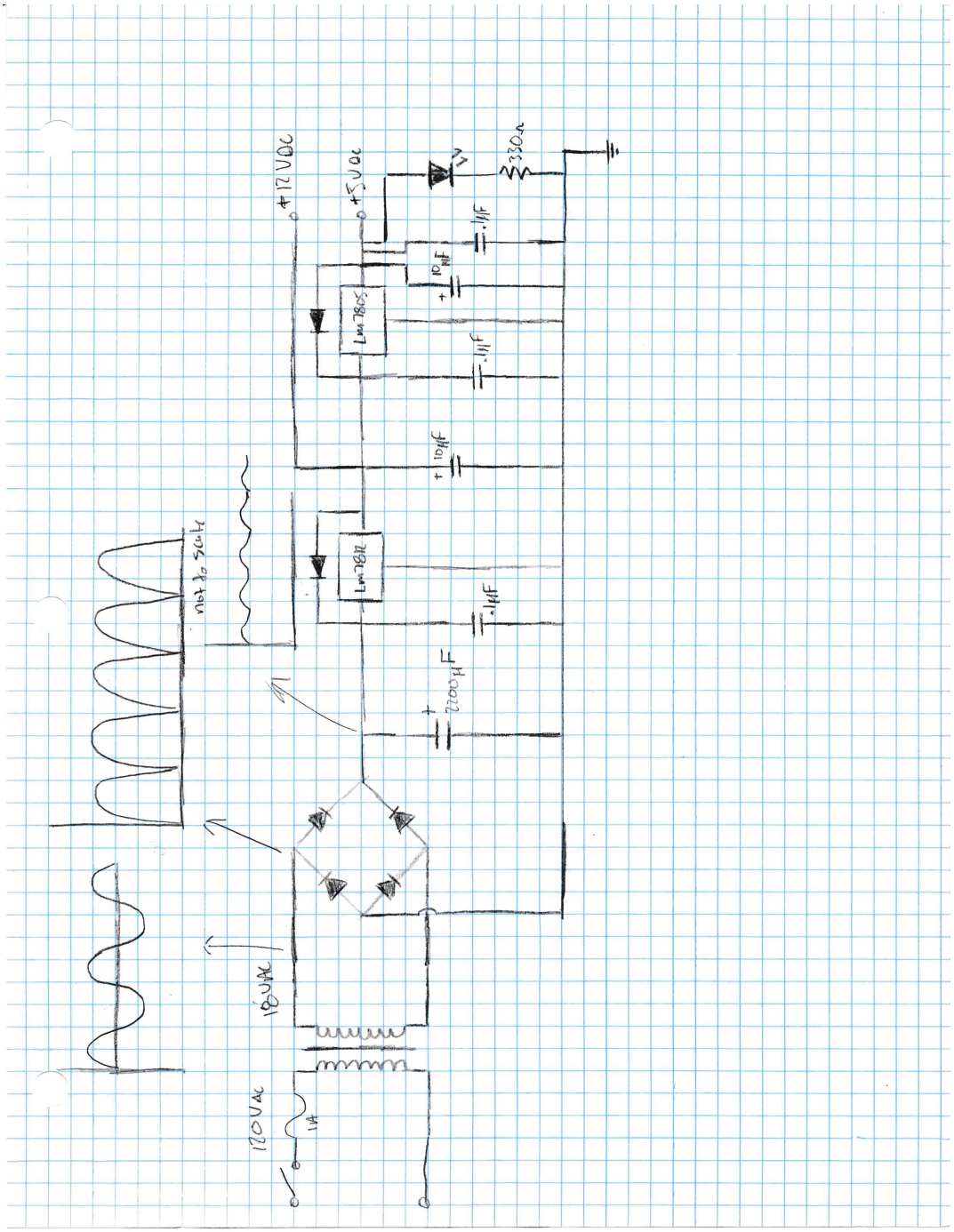
Figure 3, Rough Sketch for Water Handling Section

Item C. Schematic for 16 Relay Module



Item D. Power Supply PCB layout.

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine



Item E. Bill of Materials.

Bill of Materials				
Item	Quantity	Description	Price	Function

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine

1	4	6'x1.25" 12 awg angle iron	50.96	chassis frame
2	7	4'x1.25" 12 awg angle iron	68.39	chassis frame and shelf mounts
3	4	24"x24" .75" plywood board	22.32	shelving material
4	1	36"x24" 1/8" plexiglass panel	25.99	front panel, holds UI and other elements
5	1	24"x24" 1/8" plywood board	5.99	shelving material
6	4	DFRobot Fit0492-A 50 rpm motor	49.04	movement motors
7	2	12VDC fluid pumps	Donated	inlet and outlet water
8	1	Keurig water heating core	Donated	heats water
9	3	servo motors	39.90	controls feed
10	3	50"x4.25" pvc tube	16.79	feed/stock tubes
11	11	micro limit switches	12.99	position tracking, movement tracking, stock input
12	1	EVGA 700W Atx power supply	59.99	primary power supply
13	1	Sainsmart 16 relay module	17.66	relays
14	1	Arduino Mega 2560	36.47	microcontroller
15	1	LCD with I2C backpack	12.99	UI LCD
16		wires, hardware, screws, fasteners	75.00	various hardware and assembly items
Power Supply Components				
1	1	120 to 18 VAC stepdown trans.	17.99	step down transformer
2	1	bridge rectifier	1.09	bridge rectifier for filtering
3	1	2200 microFarad filter capacitor	2.79	main filter capacitor
4	2	10 microFarad capacitors	1.36	filtering after the linear rectifiers
5	3	.1 microFarad capacitor	1.44	trimmer capacitors
6	1	LM7812	1.34	12 VDC linear regulator
7	1	LM7805	1.32	5 VDC linear regulator
8	1	bud box	5.99	housing for the power supply
9	1	13 A fuse	1.07	inline fuse for water heater
10	1	1 A fuse	0.19	fuse for the primary side of the transformer
11	1	milling custom pcb boards	22.09	milling custom boards with JLDPCB and shipping

Total	551.15

Item F. Motor Specifications.

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine

DFROBOT
DRIVE THE FUTURE

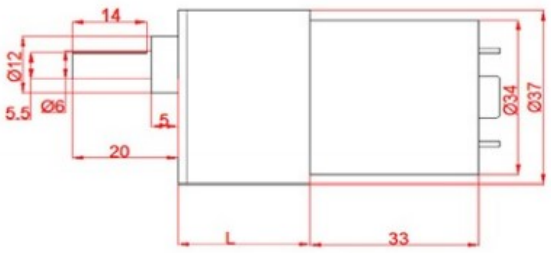
Metal DC Geared Motor - 12V 50RPM 50kg.cm

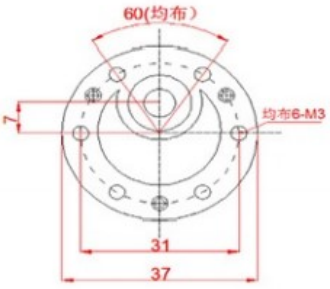
SKU:FIT0492-A


INTRODUCTION

This is a metal DC geared motor, 100% pure copper coils, high-density molecular layer, 100:1 metal reducer, small size, large torque. The maximum torque could arrive 50 kg.cm, stable and durable!

电机外形尺寸-Motor Overall Dimension





SPECIFICATION

- Rated voltage: 12 V
- Gear reduction ratio: 100:1
- D output shaft diameter: 6 mm
- No-load speed: 50 RPM @ 12 v
- No-load current: 0.17 A
- Rated speed: 45 RPM @ 12 v
- Current rating: 0.68 A

- Rated torque: 9 kg.cm
- Locked-rotor torque: 50 kg.cm
- Locked-rotor current: 2.19 A
- Power: 5W
- Weight: 210 g

SHIPPING LIST

- Metal DC Geared Motor - 12V 50RPM 50kg.cm x1

Item G. Linear Actuator Specifications.

Load (LBS)		No Load Current (A)	Full Load Current (A)	Speed (inch/sec)	
Dynamic	Static	12VDC	*12VDC*	No Load	Full Load
5	6.5	0.1	0.2	0.59	0.55

Stroke	0.5" to 12"
Limit Switch	Internal - Non-Adjustable
Limit Switch Feedback	N/A
Screw Type	ACME Screw
Motor Type	Brushed DC Motor
Connector Type	See Page 5
Wire Length	8" (customizable)
Housing Material	Aluminum Alloy
Rod Material	Stainless Steel
Gear Material	Powdered Metallurgy Steel Alloy
Color (Shaft)	Silver
Color (Motor End)	Black
Noise	<45dB
Duty Cycle	20% for < 4" or 10% for 4"-12" (20 minute period)
Operational Temperature	0ºC to 50ºC (32ºF to 122ºF)
Protection Class	IP66
Feedback Options	N/A
Certifications	CE/RoHS
Mounting Brackets	See Page 7
Mounting Ends	Standard, Non-Customizable

Item H. Operations/Setup Manual.

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine

Contents

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine

System Overview

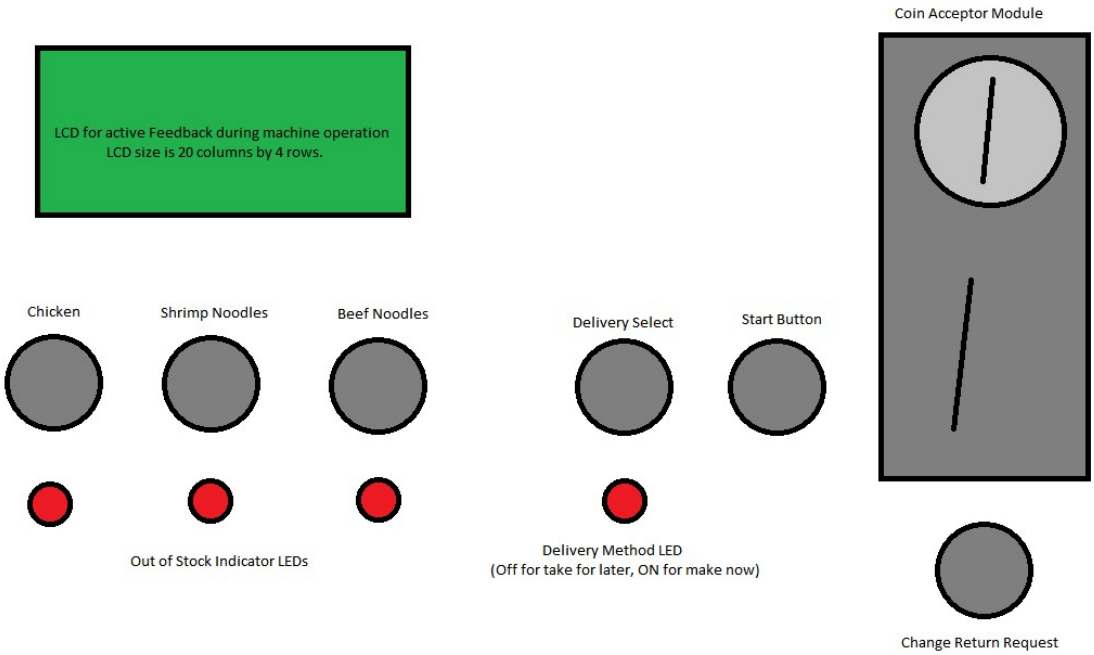
Included Hardware and Components

- Ramen Noodle Vending Machine appliance cabinet
- User Operations Guide/Manual
- Soft water line and saddle valve
- (x2) Power supply cables

Warnings and Advisories

UNDER NO CIRCUMSTANCES IS THE ARDUINO MEGA 2560 TO BE REFLASHED, THIS WILL DELETE THE OPERATING PROGRAM.
UNDER NO CIRCUMSTANCES ARE MOVEMENT COMPONENTS TO BE USER SERVICED, MUST BE RETURNED TO THE FACTORY FOR COMPONENT SERVICE.
UNDER NO CIRCUMSTANCES IS POWER TO BE TURNED OFF FROM THE MACHINE DURING OPERATION.

User Interface



Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine

Safety Considerations / Warnings

This machine uses a high power water heater. The water heater module gets very hot during normal operation and during the cleaning cycle. Use caution when working around the internal components.

……

This machine contains two stainless steel needles that are a hazard while working inside the machine. Use caution: cut resistant gloves encouraged while servicing machine.

……

When power is connected to the machine, both high current DC voltage and AC line voltage are active in the cabinet. DO NOT TOUCH ANY CONNECTIONS WHEN MACHINE IS POWERED ON! SEVERE ELECTROCUTION HAZARD!

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine

Appliance Installation

1. Unpack appliance, all installation hardware and other components.
2. Remove protective film from cabinet.
3. While near cabinets' final position, connect the soft water line to the inlet pump on the cabinet.
4. On the other end of the water inlet line, install the saddle valve on the facility water line where available.
5. Connect and secure the water line to the saddle valve.
6. Check for leaks.
7. Connect a power supply cable to each power supply.
8. Turn the power supplies on via the rocker switch on each power supply unit.

HIGH VOLTAGE AND CURRENT POWER NOW PRESENT IN CABINET! USE CAUTION!

9. Flip the master power to the cabinet to the on position.
10. Observe the cabinet lighting, LCD, and relay board in the bottom of the cabinet.
11. If one or more components are not active refer to the troubleshooting guide for more assistance.
12. Move cabinet into final position and re-ensure no leaks have formed around the water lines.
13. Machine is now ready for operation.

Operations

Standard Operation

By default, the machine starts in normal operation. This is the operation mode that allows for customer input, selection, and noodle processing.

Change Acceptance

Insert quarters (Quarters ONLY) into change acceptor, located on the right side of the user interface cluster. If the coin does not reappear at the bottom of the change acceptor, the coin has been accepted. On the LDC, located on the top left of the UI cluster, credit should appear and reflect the amount of change inserted into the machine. Machine will inform you when you have inserted adequate change. Once correct change is provided, flavor selection will be allowed.

Flavor Selection

Following change selection, the LCD will prompt selection a flavor. The flavor selection buttons are located just below the LCD. Pressing one of the buttons will update the LDC to reflect the selection, if it's in stock, and show the delivery method selected (By default the delivery is to make the noodles now). An audible tone will also play to reflect the selection. If the selection is out of stock, a different tone will play and the LCD will inform you that the selection is out of stock.

Stock Indication

If a selected flavor is out of stock, the LDC will inform you that it is out of stock. An audible tone will also inform you it is out of stock. Located directly below the selection button is a red LED that will illuminate is a flavor is out of stock.

That light will remain illuminated until that flavor is restocked. If a selected flavor is not available, it will not allow you to begin noodle processing.
If no stock is available in the machine, the machine will reject user input and reject change input and the LCD will reflect that there is no stock available.

Delivery Selection and Indication

Directly to the right of the flavor selectin buttons, the delivery select button allows selection between having noodles prepared now or if the noodles are to be dispensed dry to be prepared later. Below the delivery select button is a red LED that will indicate delivery choice, as well as being displayed on the LCD. If the LED is illuminated, the noodles will be prepared ready to eat, if the LED is no illuminated it will dispense dry noodles. By default the machine prepares the noodles immediately.

Selection Confirmation

After flavor selection and delivery selection, directly to the right of the delivery select button and indicator, the start push button will commence preparation or delivery of noodles. Once pressed, the LCD will update to reflect machine status and activity. After pressing the start button, noodles will be dispensed at the front of the machine in approximately 2 minutes.

Change Return Request

At any time before confirming selection and starting noodle preparation, change may be requested back. The machine will dispense exact change and return to the start of operation. After the start button is pressed, the change return button will not dispense change, the product is already being processed through the machine.

Automated Movement

After the start button is pressed, no more user input is necessary for machine to dispense and make noodles.

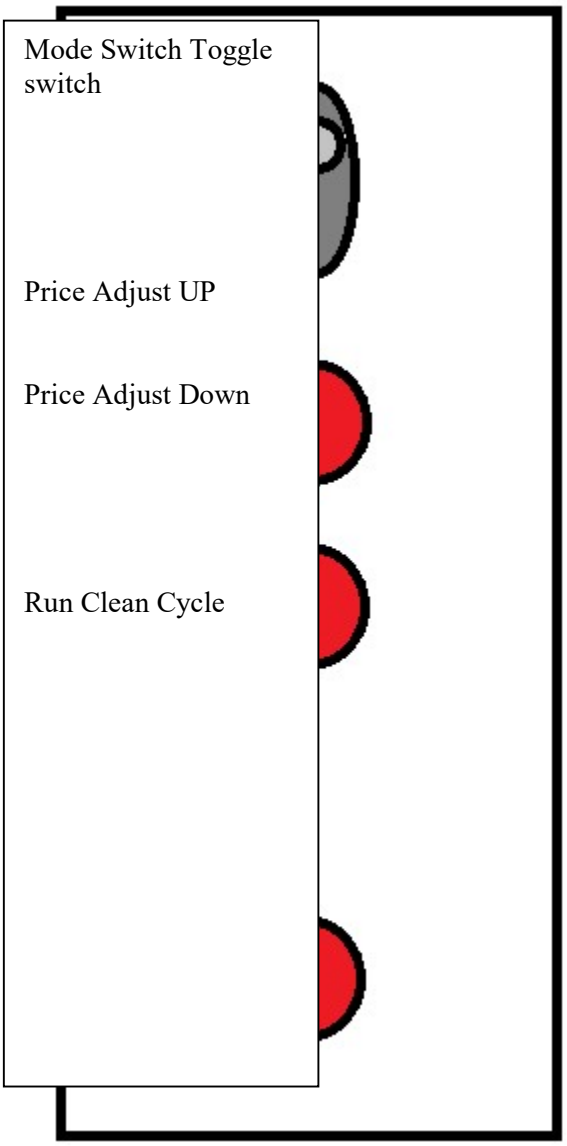
Delivery

After the machine has prepared noodles, either ready to eat or to take for later, the machine will present noodles at the front of the cabinet in the delivery window. An audible tone will play to let the user know that their noodles are ready for pickup. The LCD will also provide a message that noodles are ready for pickup. After a moment, the machine will return to the beginning of operation, ready for another customer.

Maintenance Mode Operation

When looking at the front of the machine, on the left side of the machine is the MX Mode UI panel. The toggle switch at the top of the MX Mode panel allows for switching between standard operation and MX mode operation. This UI panel is intended for service technicians and authorized personnel only, not for customer access.

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine



Mode Switching

At the top of the MX mode UI panel is a toggle switch. in the down position, the appliance will be in the standard operation mode. When flipped to the down position, the machine will transition to MX mode which will allow some specialized functions. The machine will remain in MX mode if the toggle switch is in the MX mode position.

Price Adjustment

While in MX mode, two push buttons on the side panel allow for price per unit adjustment. The button on top will increase price by .25 cent increments, price

adjustment is variable from 50 cents up to $2.50. per noodle cup. The button on the bottom will decrease price per unit by .25 cent increments.
Price adjustments will be displayed on the LCD on the front panel. Price is defaulted to .75 cents per cup on initial setup or upon program reset.

Cleaning Cycle

Before running a cleaning cycle, ensure that a bucket or bottle is placed below the puncture tip. This will avoid water getting into the mechanism or electrical components. Below the price adjust buttons is a push button that will run a cleaning cycle. In order to run a cleaning cycle, turn off water from the water line and use the bypass waterline to run cleaning/descaling solution into the machine. The cleaning cycle automatically runs pumps and heater for a period of 3 minutes. It is recommended that the cleaning cycle is run a couple of times, with a rinse clean cycle with clean water to ensure no descaling solution gets into food product.
If the machine is switched to standard operation during a cleaning cycle it will complete the cleaning cycle before changing modes.
LCD will inform when the cleaning cycle is running and when it is complete.

Re-Stocking

Restocking is only permitted during MX mode operation. Any cups added during standard op will not be counted in the stock count. The LCD will display current stock count for each tube. Insert cups, one at a time, into the top of the tubes. Do not insert more than one at a time. Sensors in the tubes will count cups as they are inserted.

Troubleshooting Guide

LCD is not active. Cabinet lights not on.

Ensure that power cables are connected to each power supply. Ensure that both power supply switches are on and master power is on.

Coin acceptor won't take coins.

Coin acceptor is programmed to only accept US currency Quarter dollar coins. The coin acceptor can be factory configured to accept other coins. If the coin acceptor is not taking coins, ensure that the appliance is no in MX mode. The machine will not take coins while in MX mode.

Power failure during operation.

If power is turned off to the machine during either mode of operation, the machine operation will be faulted and requires manual reset by a technician.

Notes

Item I. Arduino wiring list.

I/O Slot	Connected Device	Description

0	Change Input	Serial Receive from change input module
1		
2	Delivery Select	toggles between delivery now/ take for later
3	Prep Start	Begins noodle prep/delivery
4	Clean Start	Starts a cleaning cycle when in MX mode
5	Change return Rqst	Request change return from machine
6	Delivery Indicator	LED on UI panel for delivery choice indication
7	Stock Input 2	Limit switch to count cups input to stock 2
8	Stock Input 1	Limit switch to count cups input to stock 1
9	Stock Input 0	Limit switch to count cups input to stock 0
10		
11	Stock Indicator 2	LED on UI panel to show if stock 2 is out
12	Stock Indicator 1	LED on UI panel to show if stock 1 is out
13	Stock Indicator 0	LED on UI panel to show if stock 0 is out
14	price adjust down	push button to adjust price down 25 cents
15	price adjust up	push button to adjust price up 25 cents
16	mode select	toggle switch to switch to MX mode.
17	flavor select 2	push button on UI to select flavor 2
18	flavor select 1	push button on UI to select flavor 1
19	flavor select 0	push button on UI to select flavor 0
20		
21		
23	change inhibit	signal to coin unit to reject coins
24	inlet water	relay control for the inlet pump
25	outlet water	relay control for the outlet pump
26	main motor fwd	relay control for the main motor ccw direction
27	main motor rev	relay control for the main motor cw direction
28	main motor ground	relay control for switching ground.
29	feed motor 0	relay control for feed motor 0
30	feed motor 1	relay contorl for feed motor 1
31	feed motor 2	relay control for feed motor 2
32	actuator down	relay control for actuator movement down
33	actuator up	relay control for actuator movement up
34	actuator ground	relay control for actuator ground switching
35	home position	limit switch for positioning at delivery area

Comprehensive Senior Design Project Report: Ramen Noodle Vending Machine

36	water heater	relay control for water heater
37	water handle pos	limit switch for position at water handling
38	tube 2 pos	limit switch for positioning at tube 2
39	tube 1 pos	limit switch for positioning at tube 1
40	tube 0 pos	limit switch for positioning at tube 0
41	motor switch 2	rotation tracking switch on motor 2
42	motor switch 1	rotation tracking switch on motor 1
43	motor switch 0	rotation tracking switch on motor 0
44	Feed Servo 0	controls the feed servo for tube 0
45	Feed Servo 1	controls the feed servo for tube 1
46	Feed Servo 2	controls the feed servo for tube 2
47	Buzzer	buzzer for audio feedback

Item J. timeline for project development.